# Przetwarzanie rozproszone

## Plan wykładu

Celem wykładu jest zapoznanie słuchacza z podstawowymi pojęciami i problemami związanymi z przetwarzaniem rozproszonym. Wykład obejmie omówienie m. in. definicji procesu sekwencyjnego, komunikatu, kanału komunikacyjnego czy też stanu kanału. W dalszej części wykładu omówione zostaną różnego rodzaju operacje komunikacyjne, a także scharakteryzowane zostaną rodzaje komunikacji. Student zapozna się także z formalnym modelem procesu sekwencyjnego. W następnej części wykładu zdefiniowane zostanie pojęcie zdarzenia i podane zostaną definicje różnych ich rodzajów. Kolejnym tematem który zostanie poruszony będzie pojęcie warunku uaktywnienia i definicje z tym związane. Na zakończenie przedstawione zostaną klasyczne modele żądań. Omówione zostaną ich definicje formalne i własności, a także zaprezentowana zostanie w sposób graficzny ich główna idea.

## Przetwarzanie rozproszone

**Procesem rozproszonym (przetwarzaniem rozproszonym)** nazywamy współbieżne i skoordynowane (ang. *concurrent and coordinated*) wykonanie w środowisku rozproszonym zbioru $\mathcal{P}$ procesów sekwencyjnych $P_1, P_2, P_3, ..., P_n$, współdziałających w realizacji wspólnego celu przetwarzania.

## Proces sekwencyjny

Nieformalnie, każdy **proces sekwencyjny (zadanie)** jest działaniem wynikającym z wykonywania w pewnym środowisku (kontekście) **programu sekwencyjnego (algorytmu sekwencyjnego)**, który składa się z ciągu **operacji (instrukcji, wyrażeń) atomowych** (nieprzerywalnych).

## Klasy operacji

Wyróżnia się dwie podstawowe klasy operacji:

- wewnętrzne (ang. *internal, local*),
- komunikacyjne (ang. *communication*).

**Operacje wewnętrzne** odnoszą się tylko do zmiennych lokalnych programu. **Operacje komunikacyjne** odnoszą się natomiast do środowiska i dotyczą komunikatów (ang. *messages*) oraz kanałów (ang. *channels*).

## Komunikat – definicja

Formalnie, **komunikat (wiadomość)** $M = \langle tag, mId, sId, rId, data \rangle$ jest dynamiczną strukturą danych, która obejmuje na ogół następujące atrybuty:

- identyfikator typu wiadomości - $tag$,
- identyfikator wiadomości - $mId$,
- identyfikator procesu *nadawcy* (ang. *sender*) - $sId$,
- identyfikator procesu *odbiorcy* (ang. *receiver*) - $rId$,
- pole danych - $data$.

**Kanał – definicja**

**Kanał** jest obiektem (zmienną) skojarzonym z uporządkowaną parą procesów $\langle P_i, P_j \rangle$, modelującym jednokierunkowe łącze transmisyjne. Typem tego obiektu jest zbiór wiadomości, którego rozmiar nazywany jest **pojemnością kanału**.

**Kanały incydentne, wejściowe i wyjściowe**

Kanał skojarzony z parą procesów $\langle P_i, P_j \rangle$, oznaczamy przez $C_{i,j}$ oraz nazywamy kanałem *incydentnym* z procesem $P_i$ i z procesem $P_j$. Ponadto, kanał $C_{i,j}$ nazywamy **kanałem wyjściowym** procesu $P_i$ oraz **kanałem wejściowym** procesu $P_j$.

**Zbiory kanałów**

Zbiór wszystkich kanałów incydentnych procesu $P_i$ oznaczymy przez $\mathcal{C}_i$, a zbiór kanałów wejściowych i wyjściowych tego procesu odpowiednio przez $\mathcal{C}_i^{IN}$ i $\mathcal{C}_i^{OUT}$. Tak więc, $\mathcal{C}_i = \mathcal{C}_i^{IN} \cup \mathcal{C}_i^{OUT}$.

**Zbiory procesów sąsiednich**

Przyjmujemy ponadto, że:

$$\mathcal{P}_i^{IN} = \{ P_j : \langle P_j, P_i \rangle \in \mathcal{C}_i^{IN} \}$$

$$\mathcal{P}_i^{OUT} = \{ P_j : \langle P_i, P_j \rangle \in \mathcal{C}_i^{OUT} \}$$

Zbiory $\mathcal{P}_i^{IN}$ i $\mathcal{P}_i^{OUT}$ są zbiorami **procesów sąsiednich** (incydentnych), odpowiednio **wejściowych** i **wyjściowych**, procesu $P_i$.

**Stan kanału**

Przez **stan $L_{i,j}$ *kanału* $C_{i,j}$** rozumieć będziemy zbiór, lub uporządkowany zbiór, wiadomości wysłanych przez $P_i$ lecz jeszcze nie odebranych przez proces $P_j$.

**Modelowanie opóźnienia w kanale**

W celu modelowania w kanale opóźnień komunikacyjnych istotnych w perspektywie dalszych rozważań, w zbiorze wiadomości $L_{i,j}$ wyróżnia się dwa rozłączne podzbiory: zbiór wiadomości transmitowanych (ang. *in-transit*) oraz zbiór wiadomości bezpośrednio dostępnych (ang. *available, arrived, ready*). **Zbiór wiadomości transmitowanych**, oznaczony przez $\boldsymbol{L}_{i,j}^{T}$, zawiera wiadomości, które zostały wysłane przez proces $P_i$ lecz nie dotarły jeszcze do węzła realizującego proces $P_j$ i tym samym nie są dostępne (widoczne) dla $P_j$. **Zbiór wiadomości dostępnych**, oznaczony przez $\boldsymbol{L}_{i,j}^{A}$, zawiera natomiast wiadomości wysłane przez $P_i$, które dotarły już (ang. *arrived*) do węzła procesu $P_j$ i czekają w buforze kanału na ewentualne pobranie przez proces $P_j$. Oczywiście, w każdej chwili $\boldsymbol{L}_{i,j} = \boldsymbol{L}_{i,j}^{T} \cup \boldsymbol{L}_{i,j}^{A}$.

**Stan kanału – przykład**

W celu zobrazowania przedstawionych pojęć wyobraźmy sobie dwa procesy, $P_i$ oraz $P_j$, połączone kanałem komunikacyjnym $L_{i,j}$. W pewnej chwili proces $P_i$ rozpoczyna wysyłanie wiadomości do procesu $P_j$. Po pewnym czasie część z wiadomości wysłanych przez $P_i$ staje się

dostępna dla procesu $P_j$. Zwróćmy uwagę, że w kanale komunikacyjnym nadal znajdować się mogą wiadomości które zostały wysłane przez proces $P_i$, ale nie dotarły jeszcze do procesu $P_i$. Tworzą one zbiór wiadomości transmitowanych $L_{i,j}^{T}$. Wiadomości, które dotarły już do procesu $P_j$ i są dla niego dostępne, tworzą zbiór wiadomości dostępnych $L_{i,j}^{A}$. Suma obu tych zbiorów tworzy zbiór wiadomości $L_{i,j}$, wysłanych przez $P_i$ do procesu $P_j$, które nie zostały jeszcze przez niego odebrane. Powyższy slajd ilustruje przedstawione pojęcia i relacje między nimi.

**Predykaty opisujące stan kanału**

W tym kontekście, z kanałem można skojarzyć predykaty opisujące poszczególne jego stany:

- $empty(C_{i,j}) \equiv L_{i,j} = \emptyset$
- $in\text{-}transit(C_{i,j}) \equiv L_{i,j}^{T} \neq \emptyset$
- $available(C_{i,j}) \equiv L_{i,j}^{A} \neq \emptyset$

**Predykat *empty***

Predykat $empty(C_{i,j})$ zachodzi wtedy i tylko wtedy, gdy $L_{i,j} = \emptyset$ (kanał komunikacyjny jest pusty).

**Predykat *in-transit***

Predykat $in\text{-}transit\ (C_{i,j})$ zachodzi wtedy i tylko wtedy, gdy $L_{i,j}^{T} \neq \emptyset$ (zbiór wiadomości transmitowanych nie jest pusty). Innymi słowy, w kanale komunikacyjnym znajdują się wiadomości wysłane przez proces $P_i$, które nie dotarły jeszcze do procesu $P_j$.

**Predykat *available***

Predykat $available\ (C_{i,j})$ zachodzi wtedy i tylko wtedy, gdy $L_{i,j}^{A} \neq \emptyset$. Innymi słowy, istnieją wiadomości, które zostały wysłane przez proces $P_i$ i dotarły już do procesu $P_j$.

**Indywidualne operacje komunikacyjne (1)**

Zbiór podstawowych operacji komunikacyjnych obejmuje indywidualne i grupowe operacje **nadawania** (ang. *send*) i **odbioru** (ang. *receive*).

> $send(P_i, P_j, M)$ – w procesie nadawcy $P_i$, operacja ta jest sparametryzowana dodatkowo przez pojedynczą wiadomość $M$ i identyfikator procesu odbiorcy $P_j$ (lub odpowiednio kanału $C_{i,j}$). Efektem wykonania tej operacji jest umieszczenie wiadomości $M$ w kanale $C_{i,j}$, a więc wykonanie podstawienia $L_{i,j} := L_{i,j} \cup \{M\}$.

**Indywidualne operacje komunikacyjne (2)**

> $receive(P_i, P_j, inM)$ – w procesie odbiorcy $P_j$, operacja ta jest sparametryzowana przez pojedynczą zmienną $inM$ i identyfikator procesu $P_i$ – **oczekiwanego nadawcy** wiadomości. Jeżeli kanał $C_{i,j}$ nie jest pusty i pewna wiadomość $M$ jest bezpośrednio dostępna ($available(C_{i,j})$ ma wartość $True$), to efektem wykonania tej operacji jest pobranie wiadomości $M$ z kanału $C_{i,j}$, a więc wykonanie podstawienia $L_{i,j} := L_{i,j} \setminus \{M\}$ oraz $inM := M$.

**Grupowe operacje komunikacyjne (1)**

***send*($P_i$, $\mathcal{P}_i^R$, $M$)** – w procesie nadawcy $P_i$, operacja ta jest sparametryzowana przez pojedynczą wiadomość $M$ i zbiór procesów $\mathcal{P}_i^R \subseteq \mathcal{P}_i^{OUT}$, będących odbiorcami (adresatami) wiadomości $M$. Efektem wykonania tej operacji jest umieszczenie wiadomości we wszystkich kanałach $C_{i,j}$, takich że $P_j \in \mathcal{P}_i^R$, a więc podstawienie dla wszystkich tych kanałów: $L_{i,j} := L_{i,j} \cup \{M\}$.

**Grupowe operacje komunikacyjne (2)**

***receive*($\mathcal{P}_j^S$, $P_j$, *sInM*)** – w procesie odbiorcy $P_j$ operacja ta jest sparametryzowana przez zmienną $sInM$ typu zbiór wiadomości i zbiór procesów $\mathcal{P}_i^S \subseteq \mathcal{P}_j^{IN}$, będących **oczekiwanymi nadawcami** wiadomości. W ogólności, warunkiem wykonania tej operacji jest jednoczesna dostępność wiadomości od wszystkich procesów $P_i \in \mathcal{P}_j^S$. Jeżeli warunek ten jest spełniony, to efektem wykonania operacji $receive(\mathcal{P}_j^S, P_j, sInM)$ jest atomowe pobranie wiadomości $M_i$ od procesów $P_i \in \mathcal{P}_j^S$ i umieszczanie ich w $sInM$. Tym samym, dla każdego procesu $P_i \in \mathcal{P}_j^S$, wykonywane jest kolejno podstawienie $L_{i,j} := L_{i,j} \setminus \{M_i\}$ oraz $sInM := sInM \cup \{M_i\}$.

**Rodzaje komunikacji**

Kanały o niezerowej pojemności umożliwiają realizację nieblokowanych (asynchronicznych) i blokowanych (synchronicznych) operacji komunikacji. W **operacjach komunikacji nieblokowanej**, proces nadający przekazuje komunikat do kanału i natychmiast kontynuuje swoje działanie, a proces odbierający - odczytuje stan kanału wejściowego, lecz nawet gdy kanał jest pusty, proces kontynuuje działanie. W wypadku operacji **komunikacji blokowanej**, nadawca jest wstrzymywany do momentu, gdy wiadomość zostanie odebrana przez adresata, natomiast odbiorca - do momentu, gdy oczekiwana wiadomość pojawi się w jego buforze wejściowym. W tym kontekście, wyróżnia się komunikację synchroniczną i asynchroniczną.

**Komunikacja synchroniczna**

W **komunikacji synchronicznej**, nadawca jest blokowany, aż odpowiedni odbiorca odczyta przesłaną do niego wiadomość, a odbiorca jest blokowany, aż nadana wiadomość do niego dotrze (*rendez-vous*).

**Komunikacja asynchroniczna**

W przypadku **komunikacji asynchronicznej**, nadawca lub odbiorca komunikuje się w sposób nieblokowany.

W dalszej części pracy typ operacji będzie wynikać z kontekstu lub domyślnie przyjmować będziemy, że stosowane są… operacje asynchronicznego nadania i synchronicznego odbioru. Jeśli okaże się to jednak niezbędne, to dla odróżnienia asynchroniczne operacje odbioru będziemy oznaczać przez ***receive_a***($P_i$, $P_j$, $inM$) lub ***receive_a***($\mathcal{P}_j^S$, $P_j$, $sInM$), a synchroniczne operacje nadania - przez ***send_s***($P_i$, $P_j$, $M$) lub ***send_s***( $P_i$, $\mathcal{P}_i^R$, $M$ ).

**Model formalny procesu sekwencyjnego**

Formalnie, proces sekwencyjny może być opisany (modelowany) przez uporządkowaną czwórkę $P_i = \langle \mathcal{S}_i, \mathcal{S}_i^0, \mathcal{E}_i, \mathcal{F} \rangle$, gdzie

$\mathcal{S}_i$ – jest **zbiorem stanów** procesu $P_i$,

$\mathcal{S}_i^0$ – jest **zbiorem stanów początkowych**, $\mathcal{S}_i^0 \subseteq \mathcal{S}_i$,

$\mathcal{E}_i$ – jest **zbiorem zdarzeń** procesu ,

$\mathcal{F}$ – jest **funkcją tranzycji**, taką że: $\mathcal{F}_i \subseteq \mathcal{S}_i \times \mathcal{E}_i \times \mathcal{S}_i$, a $\langle S, E, S' \rangle \in \mathcal{F}_i$ , jeżeli zajście zdarzenia $E$ w stanie $S$ jest możliwe i prowadzi do zmiany stanu na $S'$.

**Stan procesu (1)**

***Stan*** $S_i(t)$ ***procesu*** w chwili czasu lokalnego jest w ogólności zbiorem wartości wszystkich zmiennych lokalnych skojarzonych z procesem w chwili oraz ciągów wiadomości wysłanych (wpisanych) do incydentnych kanałów wyjściowych i ciągów wiadomości odebranych z incydentnych kanałów wejściowych do chwili $t$.

**Stan procesu (2)**

Dla każdego $t$, $S_i(t) \in \mathcal{S}_i$. W celu uproszczenia notacji, zależność stanu od czasu można przyjąć za domyślną i jeśli nie prowadzi to do niejednoznaczności, oznaczać stan w pewnej chwili $t$ przez $S_i$.

Zbiór $\mathcal{S}_i^0$ jest **zbiorem stanów początkowych**, których wartości są zadawane wstępnie, bądź są wynikiem zajścia wyróżnionego **zdarzenia inicjującego** $E_i^0$.

**Definicja zdarzenia**

***Zdarzenie*** $E_i^k$ odpowiada unikalnemu wykonaniu operacji atomowej, zmieniającemu stan $S_i$ procesu i ewentualnie stan incydentnych z procesem kanałów $C_{i,j}$ lub $C_{j,i}$. Jeżeli operacja odpowiadająca zdarzeniu została wykonana, to powiemy, że **zdarzenie zaszło**.

**Klasy zdarzeń**

Wyróżnia się trzy podstawowe klasy zdarzeń: $e\_send$, $e\_receive$ i $e\_internal$.

**Zdarzenie *e_send***

W szczególności, zdarzenie $e\_send(P_i, P_j, M)$ zachodzi w procesie $P_i$, w wyniku wykonania przez ten proces operacji $send(P_i, P_j, M)$. Analogicznie definiuje się zdarzenie $e\_send(P_i, \mathcal{P}_i^R, M)$.

**Zdarzenie e_receive**

Zdarzenie $e\_receive(P_i, P_j, M)$ zachodzi natomiast w procesie $P_j$, gdy $P_j$ wykonał operację $receive(P_i, P_j, inM)$, a odczytana do zmiennej lokalnej $inM$ wiadomość $M$ pochodziła od procesu $P_i$. Analogicznie definiuje się zdarzenie $e\_receive(\mathcal{P}_j^S, P_j, \mathcal{M}_j^S)$, odpowiadające wykonaniu operacji Zbiór $\mathcal{M}_j^S$ jest tu zbiorem jednocześnie odbieranych wiadomości z kanałów wejściowych.

**Zdarzenie e_internal**

Powiemy, że w procesie $P_i$ zaszło zdarzenie **lokalne** $e\_internal(P_i)$, gdy proces ten wykonał operację, która nie zmienia stanu kanałów incydentnych procesu. Do zdarzeń lokalnych zalicza się między innymi zdarzenia: $e\_init(P_i,S_i^k)$ – które nadaje procesowi $P_i$ stan $S_i^k$ (w szczególności stan początkowy $S_i^0$), oraz zdarzenie $e\_stop(P_i)$ – które kończy wykonywanie procesu $P_i$.

**Dostępność wiadomości**

Należy zaznaczyć, że zdarzenie $e\_send(P_i,P_j,M)$ odpowiada na ogół wykonaniu operacji asynchronicznej, która zapisuje wiadomość $M$ do kanału wyjściowego (bufora łącza transmisyjnego). Następnie środowisko rozproszone transmituje wiadomość $M$ do węzła docelowego i zapisuje ją w buforze wejściowym procesu docelowego $P_j$. Gdy wiadomość dotrze do bufora wejściowego procesu docelowego, to wiadomość staje się dostępna i może być natychmiast pobrana z bufora w wyniku wykonania operacji $receive(P_i,P_j,inM)$. Dostępność wiadomości utożsamiać też można z zajściem zdarzeń w środowisku komunikacyjnym: **zdarzenia dostarczenia wiadomości** $e\_deliver(P_i,P_j,M)$, albo **zdarzenia nadejścia wiadomości** – $e\_arrive(P_i,P_j,M)$. W tym kontekście, przez $\mathcal{P}_j^A$ oznaczymy zbiór procesów, których wiadomości dotarły i są dostępne dla $P_j$. Jeżeli proces odbiorcy $P_j$ odczytuje skierowaną do niego wiadomość $M$, wykonując operację $receive(P_i,P_j,inM)$, wiadomość ta jest przepisywana (przemieszczana) z bufora wyjściowego łącza (bufora wejściowego procesu $P_i$) do lokalnej zmiennej procesu $inM$. Jeżeli przepisanie to nastąpiło, to mówimy, że wiadomość **została odebrana** lub, że zaszło **zdarzenie odbioru** $e\_receive(P_i,P_j,inM)$.

Warto przypomnieć, że wiadomości mają swoje unikalne identyfikatory, które pozwalają na jednoznaczne kojarzenie zdarzenia nadania konkretnej wiadomości ze zdarzeniem odbioru tej samej wiadomości.

**Funkcja tranzycji**

**Funkcja tranzycji** $\mathcal{F}_i \subseteq \mathcal{S}_i \times \mathcal{E}_i \times \mathcal{S}_i$ opisuje reguły zmiany stanu $S$ na $S'$ w wyniku zajścia zdarzenia $E$. Elementy $\langle S,E,S'\rangle \in \mathcal{F}_i$ nazwiemy **tranzycjami** lub **krokami**. W zależności od zachodzącego zdarzenia $E$, tranzycję nazwiemy odpowiednio **tranzycją wejścia**, **wyjścia** lub **lokalną**

**Zdarzenia dopuszczalne**

Funkcja tranzycji dopuszcza możliwość zajścia zdarzenia $E$ tylko w tych stanach $S$, dla których $\langle S,E,S'\rangle \in \mathcal{F}_i$. Dlatego też, w wypadku gdy $\langle S,E,S'\rangle \in \mathcal{F}_i$, powiemy że zdarzenie $E$ jest **dopuszczalne** (ang. *allowed*) w stanie $S$. Wprowadzimy też predykat $allowed(E)$ oznaczający, że w danej chwili zdarzenie $E$ jest dopuszczalne.

**Zdarzenia gotowe**

Oprócz czynnika wewnętrznego (stanu procesu), zajście zdarzenia może być dodatkowo uwarunkowane stanem kanałów wejściowych (środowiska). Na przykład, zdarzenie $e\_receive(P_i,P_j,M)$ może zajść tylko wówczas, gdy wiadomość $M$ została wcześniej wysłana przez proces $P_i$ i jest aktualnie dostępna w kanale $C_{i,j}$. Podobnie, zdarzenie $e\_receive(\mathcal{P}_j^S,P_j,\mathcal{M}_j^S)$ może zajść tylko wówczas, gdy jest spełniony warunek wykonania operacji $receive(\mathcal{P}_j^S,P_j,sInM)$. Jeżeli zdarzenie może zajść ze względu na warunki zewnętrzne (stan kanałów), to powiemy że zdarzenie jest **przygotowane** lub **gotowe** (ang. *ready*). Fakt gotowości zdarzenia w danej chwili wyrażać będzie predykat $ready(E)$.

**Predykat *enable***

W tym kontekście wprowadzimy jeszcze predykat $enable(E)$, oznaczający, że zdarzenie jest **aktywne** – czyli jednocześnie gotowe i dopuszczalne. Stąd też:

$$enable(E) \equiv ready(E) \wedge allowed(E)$$

**Procesy zakończone, wstrzymane**

Powiemy, że proces jest w **stanie końcowym** $S_i^e$, jeżeli zbiór zdarzeń dopuszczalnych w tym stanie jest pusty. Jeżeli natomiast niepusty zbiór zdarzeń dopuszczalnych zawiera wyłącznie zdarzenia odbioru i żadne z tych zdarzeń nie jest aktywne (gotowe), to powiemy że proces jest **wstrzymany** (**zablokowany**).

**Procesy aktywne, pasywne**

Proces wstrzymany lub zakończony nazwiemy **pasywnym**. Przez proces **aktywny** będziemy natomiast rozumieć proces, który nie jest pasywny.

Przyjmijmy jeszcze, że w każdej chwili $t$ stan procesu $P_i$ reprezentuje zmienna logiczna $passive_i$, przyjmująca wartość $True$, gdy proces $P_i$ jest pasywny, a wartość $False$, gdy proces ten jest aktywny.

**Proces aktywny**

W wielu praktycznych zastosowaniach, do analizy interesujących nas właściwości procesów wystarcza uproszczony model, w którym wyróżnia się tylko dwa stany procesów: aktywny i pasywny. **Aktywny** proces $P_i$ ($passive_i$=$False$) może wysyłać i odbierać wiadomości, wykonywać tranzycje lokalne, a więc potencjalnie może również spontanicznie (w dowolnej chwili) zmienić swój stan na pasywny.

**Proces pasywny**

W stanie **pasywnym** procesu $P_i$ ($passive_i$=$True$) dopuszczalne są natomiast co najwyżej zdarzenia odbioru. Stąd zmiana stanu procesu z pasywnego na aktywny uwarunkowana jest osiągnięciem gotowości przez choćby jedno z dopuszczalnych zdarzeń odbioru, czyli spełnieniem tak zwanego warunku uaktywnienia.

**Warunek uaktywnienia**

Warunek uaktywnienia (ang. *activation condition*) $P_i$ związany jest ze zbiorem warunkującym $\mathcal{D}_i$, zbiorem $\mathcal{P}_i^A$, oraz predykatem $activate_i(\mathcal{X})$.

**Zbiór warunkujący**

Zbiór warunkujący (ang. *dependent set*), jest sumą mnogościową zbiorów $\mathcal{P}_i^S$ wszystkich zdarzeń odbioru dopuszczalnych w danej chwili.

**Predykat activate**

Z kolei predykat $activate_i(\mathcal{X})$ zdefiniowany jest w sposób następujący:

1. jeżeli $\mathcal{X} = \mathcal{D}_i$, to $activate_i(\mathcal{X}) = True$
2. jeżeli $\mathcal{X} = \varnothing$, to $activate_i(\mathcal{X}) = False$
3. jeżeli $\mathcal{X} \subset \mathcal{D}_i$ i $\mathcal{X} \neq \varnothing$, to:

   $$activate_i(\mathcal{X}) \equiv \exists \mathcal{X}' :: \mathcal{X}' \neq \varnothing \wedge \mathcal{X}' \subseteq \mathcal{X} \wedge (\mathcal{P}_i^A = \mathcal{X}' \Rightarrow (passive_i \rightsquigarrow \neg passive_i))$$

   gdzie $passive_i \rightsquigarrow \neg passive_i$ oznacza, że pasywny proces $P_i$ zmieni swój stan na aktywny w skończonym choć nieprzewidywalnym czasie.

Predykat $activate_i(\mathcal{X})$ zachodzi zatem wtedy i tylko wtedy, gdy w związku z nadejściem (dostępnością) wiadomości od procesów tworzących zbiór $\mathcal{X}' \subseteq \mathcal{X}$ ($\mathcal{X}' = \mathcal{P}_i^A$), gotowe stanie się którekolwiek z dopuszczalnych zdarzeń odbioru. Innymi słowy predykat $activate_i(\mathcal{X})$ zachodzi, jeżeli dostarczenie wiadomości od wszystkich procesów $P_j \in \mathcal{X}$ umożliwia uaktywnienie procesu $P_i$. Jak łatwo zauważyć, predykat ten posiada właściwość monotoniczności: jeżeli $\mathcal{X} \subseteq \mathcal{Y}$ i $activate_i(\mathcal{X}) = True$, to $activate_i(\mathcal{Y}) = True$ .

**Predykat *ready***

W tym kontekście, **warunek uaktywnienia** procesu może być formalnie wyrażony przez następujący predykat $ready_i(\mathcal{X})$:

$$ready_i(\mathcal{X}) \equiv (\mathcal{P}_i^A \supseteq \mathcal{X}) \wedge activate_i(\mathcal{X})$$

Gdy proces jest uaktywniany, to wiadomości, których dostarczanie doprowadziło do spełnienia warunku uaktywnienia, są atomowo pobierane z buforów wejściowych i dalej przetwarzane.

**Modele żądań**

W praktyce, warunki uaktywnienia wyrażane są często w kategoriach tak zwanego *modelu żądań* (ang. *request model*), pozwalającego na zwięzłe sformułowanie warunków uaktywnienia specyficznych dla określonej klasy zastosowań. Podejście takie odpowiada rozwiązaniom, w których procesy w trakcie wykonywania kierują do innych procesów żądania przydziału ogólnie rozumianych zasobów, lub podjęcia określonych działań. Spełnienie żądania może być potwierdzone przez przesłanie stosownej wiadomości. Tym samym, istotą modelu żądań jest wysyłanie żądań w formie wiadomości i oczekiwanie na określony zbiór wiadomości potwierdzających realizację żądania.

**Model *jednostkowy***

W **modelu jednostkowym** warunkiem uaktywnienia pasywnego procesu $P_i$ jest przybycie wiadomości od jednego, ściśle określonego nadawcy. W tym przypadku $|\mathcal{D}_i| = 1$, dla każdego naturalnego $i$, $1 \leqslant i \leqslant n$. Model ten odpowiada szerokiej klasie systemów, w których procesy żądają kolejno po jednym tylko zasobie. Przykładowo może to być przydział konkretnego bufora jednostkowego w kolejnym węźle sieci komunikacyjnej z komutacją pakietów, czy potwierdzenie założenia blokady na pojedynczym obiekcie (relacji) w rozproszonej bazie danych.

**Model *AND***

W **modelu AND** proces pasywny staje się aktywnym, jeżeli dotarły wiadomości od wszystkich procesów tworzących zbiór warunkujący. Model ten nazywany jest również **modelem zasobowym**. Jest on stosowany w rozwiązaniach, w których procesy ubiegają się o pewne wieloelementowe zbiory zasobów. Przykładowo, mogą to być konkretne bufory w kolejnych węzłach sieci komunikacyjnej tworzących kanał wirtualny, czy też blokady związane jednocześnie z wieloma obiektami.

**Model *OR***

W **modelu OR** do uaktywnienia procesu wystarczy jedna wiadomość od któregokolwiek z procesów ze zbioru warunkującego. Model ten nazywany jest również **modelem komunikacyjnym**. Odpowiada on alternatywnym strukturom programowym dostępnym między innymi w językach *CSP* i *ADA*, a także – ubieganiu się o zasoby z pewnej puli zasobów równoważnych (alternatywnych). Przykładowo, zasobami takimi mogą być pojedyncze bufory z puli buforów dostępnych w kolejnym węźle, czy też repliki obiektów. Wówczas, dostęp do jednego z zasobów z określonej puli (dostęp do jednej repliki) jest wystarczający dla kontynuowania procesu.

**Podstawowy model *k spośród r***

W **podstawowym modelu k spośród r**, z pasywnym procesem $P_i$ skojarzony jest zbiór warunkujący $\mathcal{D}_i$, liczba naturalna $k_i$, $1 \leqslant k_i \leqslant |\mathcal{D}_i|$, oraz liczba $r_i = |\mathcal{D}_i|$. W modelu tym proces staje się aktywny tylko wówczas, gdy uzyska wiadomości od co najmniej $k_i$ różnych procesów ze zbioru warunkującego $\mathcal{D}_i$.

Model ***k spośród r*** jest uogólnieniem modelu AND oraz modelu *OR*. Istotnie, model ten redukuje się do modelu AND, gdy $k_i = r_i$ , a do modelu *OR*, gdy $k_i = 1$. Model k spośród r, odpowiada na przykład ubieganiu się o bufory w sieciach komunikacyjnych, w których przesyłane wiadomości składają się z wielu ($k$) pakietów. Jeżeli w tym wypadku pojedynczy pakiet pamiętany może być w buforze jednostkowym, to przesłanie wiadomości wymaga zarezerwowania w następnym węźle puli $k$ równoważnych buforów z ogólnej ich liczby $r$, w celu przesłania do nich wiadomości składającej się z $k$ pakietów. Model *k spośród r* wykorzystywany jest również w systemach replikowanych baz danych, w których stosowany jest algorytm głosowania do zarządzania współbieżnością (ang. *quorum based concurrency control algorithm*). W algorytmach tych transakcje realizujące odczyt muszą odczytać $k^r$ kopii spośród $r$, a transakcje realizujące zapis, muszą uzyskać dostęp do $k^w$ kopii spośród $r$, przy czym $k^r + k^w > r$ oraz $2\,k^w > r$.

**Podstawowy model *k spośród r – przykład***

Na slajdzie ilustrującym model *k spośród r* widzimy przykład, w którym do aktywacji procesu wystarczą wiadomości od dwóch z pięciu procesów należących do zbioru warunkującego.

**Model *OR-AND***

W *modelu* **OR-AND** zbiór warunkujący $\mathcal{D}_i$ pasywnego procesu $P_i$ jest zdefiniowany jako $\mathcal{D}_i^1 \cup \mathcal{D}_i^2 \cup ... \cup \mathcal{D}_i^{qi}$, gdzie dla każdego naturalnego $u$, $1 \leqslant u \leqslant q_i$, $\mathcal{D}_i^u \subseteq \mathcal{P}$. Proces staje się aktywny po

otrzymaniu wiadomości od każdego z procesów tworzących zbiór $\mathcal{D}_i^1$ lub od każdego z procesów tworzących zbiór $\mathcal{D}_i^2$ lub ... lub od każdego z procesów tworzących zbiór $\mathcal{D}_i^{qi}$.

**Model *OR-AND* – przykład**

Rozważmy przykładowy warunek uaktywnienia procesu $P_i$, sformułowany następująco: proces $P_i$ zostanie uaktywniony po otrzymaniu wiadomości jednocześnie od procesów $P_1$ i $P_2$, lub jednocześnie od procesów $P_3$, $P_4$ i $P_5$, lub jednocześnie od procesów $P_6$, i $P_7$.

W tym wypadku możemy zatem przyjąć, że:

$\mathcal{D}_i^1=\{P_1, P_2\}$

$\mathcal{D}_i^2=\{P_3, P_4, P_5\}$

$\mathcal{D}_i^3=\{P_6, P_7\}$

Model OR-AND jest ogólniejszy od podstawowego modelu *k spośród r*. Model ten redukuje się do modelu *k spośród r*, gdy podzbiory $\mathcal{D}_i$ odpowiadają wszystkim możliwym podzbiorom o liczności $k$ zbioru warunkującego $\mathcal{D}_i$ o liczności $r$.

Przykładowo, jeżeli $\mathcal{D}_i=\{P_1,P_2,P_3\}$ i $P_i$ formułuje warunek *2 spośród 3*, czyli 2 spośród $\{P_1,P_2,P_3\}$, to $\mathcal{D}_i$ można wyrazić jako $\{P_1,P_2\}\cup\{P_2,P_3\}\cup\{P_1,P_3\}$, a więc jako sumę wszystkich dwuelementowych podzbiorów zbioru $\mathcal{D}_i$.

**Dysjunkcyjny model *k spośród r***

W **modelu dysjunkcyjnym *k spośród r*** z każdym pasywnym procesem $P_i$ skojarzony jest zbiór warunkujący $\mathcal{D}_i=\mathcal{D}_i^1\cup\mathcal{D}_i^2\cup...\cup\mathcal{D}_i^{qi}$, liczby naturalne $k_i^1, k_i^2, ..., k_i^{qi}$, oraz liczby naturalne $r_i^1, r_i^2, ..., r_i^{qi}$, gdzie dla każdego naturalnego $u$, $1\leqslant u\leqslant q_i$, $\mathcal{D}_i^u\subseteq\mathcal{P}$, $1\leqslant k_i^u\leqslant r_i^u=|\mathcal{D}_i^u|$. Proces staje się aktywny po otrzymaniu wiadomości od $k_i^1$ różnych procesów ze zbioru $\mathcal{D}_i^1$, lub $k_i^2$ wiadomości od różnych procesów ze zbioru $\mathcal{D}_i^2$, lub ... lub $k_i^u$ wiadomości od różnych procesów ze zbioru $\mathcal{D}_i^{qi}$.

**Dysjunkcyjny model *k spośród r – przykład***

Należy zauważyć, że model dysjunkcyjny *k spośród r* redukuje się do modelu *OR-AND* (gdy $k_i^u=|\mathcal{D}_i^u|$ dla każdego $u$) – i w konsekwencji do innych modeli. W istocie, model podstawowy k spośród r uzyskuje się gdy $q_i=1$; model *AND* – gdy $q_i=1$ i $k_i^1=|\mathcal{D}_i^1|$; a w końcu model *OR* - gdy $q_i=1$ i $k_i^1=1$.

**Model *predykatowy***

W **modelu predykatowym**, dla każdego pasywnego procesu $P_i$ ze zbiorem warunkującym $\mathcal{D}_i$ określony jest predykat $activate_i(\mathcal{X})$, gdzie $\mathcal{X} \subseteq \mathcal{P}$.

Jak łatwo zauważyć, stosownie definiując predykat $activate_i(\mathcal{X})$ można oczywiście uzyskać wszystkie wcześniej omówione modele żądań.

Rozważmy na początek dysjunkcyjny model *k spośród r*, w którym żądania procesu $P_i$ są zdefiniowane w kategoriach podzbiorów $\mathcal{D}_i^1$, $\mathcal{D}_i^2$, ..., $\mathcal{D}_i^{qi}$ oraz stałych naturalnych $k_i^1$, $k_i^2$, ..., $k_i^{qi}$ i $r_i^1$, $r_i^2$ , ..., $r_i^{qi}$. Wówczas, predykat można wyrazić następująco:

$$activate_i(\mathcal{X}) \equiv (\exists u :: 1 \leqslant u \leqslant q_i :: |\mathcal{D}_i^u \cap \mathcal{X}| \geqslant k_i^u)$$

Definicje predykatu dla innych modeli można wyprowadzić z powyższego sformułowania w sposób następujący:

model podstawowy *k spośród r* – przyjmując, że $q_i$=1, $\mathcal{D}_i^1 = \mathcal{D}_i$ i dalej:

$$activate_i(\mathcal{X}) \equiv |\mathcal{D}_i \cap \mathcal{X}| \geqslant k_i^1$$

model OR-AND – przyjmując, że $\forall u :: 1 \leqslant u \leqslant q_i :: k_i^u = |\mathcal{D}_i^u|$ i dalej:

$$activate_i(\mathcal{X}) \equiv (\exists u :: 1 \leqslant u \leqslant q :: \mathcal{D}_i^u \subseteq \mathcal{X})$$

model OR – przyjmując, że $\forall u :: 1 \leqslant u \leqslant q_i :: |\mathcal{D}_i^u| = 1 \wedge k_i^u = 1 \wedge \mathcal{D}_i = \mathcal{D}_i^1 \cup \mathcal{D}_i^2 \cup ... \cup \mathcal{D}_i^{qi}$ i dalej:

$$activate_i(\mathcal{X}) \equiv \mathcal{D}_i \cap \mathcal{X} \neq 0$$

model *AND* - przyjmując, że $q_i$ =1, $\mathcal{D}_i^1 = \mathcal{D}_i$, $k_i^1 = r_i^1 = |\mathcal{D}_i^1| = |\mathcal{D}_i|$ i dalej

$$activate_i(\mathcal{X}) \equiv \mathcal{D}_i \subseteq \mathcal{X}$$